Exploring the Visual Styles of Arcade Game Assets

Antonios Liapis

Institute of Digital Games, University of Malta, Msida, Malta

Abstract. This paper describes a method for evolving assets for video games based on their visuals properties. Focusing on assets for a space shooter game, a genotype consisting of turtle commands is transformed into a spaceship image composed of human-authored sprite components. Due to constraints on the final spaceships' plausibility, the paper investigates two-population constrained optimization and constrained novelty search methods. A sample of visual styles is tested, each a combination of visual metrics which primarily evaluate balance and shape complexity. Experiments with constrained optimization of a visual style demonstrate that a visually consistent set of spaceships can be generated, while experiments with constrained novelty search demonstrate that several distinct visual styles can be discovered by exploring along select, or all, visual dimensions.

1 Introduction

The design and development of digital games requires an extraordinary amount of creativity from the part of human game developers. Digital games rely on several creative domains such as visual art, sound design, interaction design, narrative, virtual cinematography, aesthetics and environment beautification [1]. All of these creative domains are fused together in a single software application, which must be engaging and entertaining to a broad range of players with diverse tastes and skillsets. While game development still remains primarily a human endeavor, there has been considerable interest both within the game industry and within academia in the automatic, algorithmic generation of game elements. Procedural content generation for games serves two purposes from the perspective of commercial game developers: (a) it alleviates much of the effort of manual asset creation as games are "going to need a lot more content" in the future [2], (b) it can increase the replayability value of the game as players constantly encounter new content in a "perpetually fresh world" [3]. While most commercial and academic applications of procedural content generation revolve around game levels or rules and evaluate their playability and tractable properties [4], there is merit in considering the visual appeal of the generated content and in developing computational creators that can exhibit creativity comparable to that of human game visual artists.

This paper describes a generator of game assets grounded on visual principles of human perception. Specifically, constrained evolution creates spaceship sprites which can be used as player-controlled avatars or waves of enemies for an arcade space shooter game such as *Ikaruga* (Treasure 2008) and *M.U.S.H.A.* (Compile 1990). Towards that end, several constraints on the plausibility of spaceships are encoded, and their satisfaction guaranteed via two-population constrained evolution. Spaceships which satisfy the constraints are evaluated on certain visual properties, mostly revolving around balance

and shape complexity [5]. These visual metrics are aggregated into objective functions which are optimized by a feasible-infeasible two-population genetic algorithm [6]; results show that evolved content can exhibit a consistent visual style as specified by a designer who combines desirable metrics into the algorithm's objective. Moreover, the same visual metrics are used to measure the difference between generated spaceships in a feasible-infeasible novelty search algorithm [7]; results show that the exploration afforded by novelty search discovers several distinct visual styles in an unsupervised fashion. The initial investigation in this paper can be expanded by including constraints and evaluations on the spaceships' in-game behavior, by expanding the range of visual metrics considered or by deriving visual quality from machine-learned models.

This paper contributes to the literature on computational game creativity [1] by fulfilling a graphic design task via evolutionary algorithms. More importantly, the experiments test how novelty search performs in constrained spaces as feasible-infeasible novelty search, and compare the impact of different distance functions on the appearance and variety of evolved assets. This constitutes a substantial contribution to constrained novelty search, which is still an under-explored topic [7, 8], and more generally to the study on novelty search. Finally, the analysis regarding both the quality and the diversity of the final evolved content is a contribution to search-based procedural content generation research, especially concerning generators' expressivity [9, 10].

2 Related Work

The evolutionary algorithms used for generating spaceships fall under the domain of constrained optimization and constrained novelty search, while the constraints and evaluations of generated content originate from a broad literature on search-based procedural content generation for games. This section highlights work on these related domains.

2.1 Procedural Content Generation

Using algorithmic methods to generate content for games has a long history within the game industry, dating back to *Rogue* (Toy and Wichman, 1980) and *Elite* (Acornsoft, 1984) which used procedural content generation (PCG) to create the game's dungeons and galaxies respectively. Currently, PCG is often used in games created by small teams in order to quickly create vast gameworlds such as those in *No Man's Sky* (Hello Games 2016) and engage players with perpetually fresh content such as new dungeons in *Torchlight* (Runic 2009) or new noble bloodlines in *Crusader Kings II* (Paradox 2012).

Academic interest in PCG for games is relatively recent but has received significant attention in the last decade, particularly for the application of artificial evolution for PCG under the umbrella term of *search-based PCG* [11]. Search-based PCG often requires carefully designed objective functions, which are often inspired by game design theory and design patterns [12], and may target player balance in multi-player games [13] or challenge in single player games [14]. Since generated content often come with playability concerns, constrained evolution has been applied to the generation of traversable game levels [14], as well as in generators used as part of the design process [15]. However, the types of content generated in search-based PCG and the

objective functions of such projects are surprisingly narrow in scope: most generators attempt to optimize game levels of some sort, while the fitness functions evaluate purely functional properties such as safety of resources and reachability of remote locations. Few search-based PCG projects aim to generate graphical assets, or evaluate them on their visual qualities: of note are the game shaders of [16] which evolved towards a designer-specified color, the weapon particles of [17] which evolved based on users' gameplay, and flowers of [18] which evolved via playful interactive evolution. Previous work by the author on spaceship generation [19] revolved around evolving the spaceships' geometry directly, and used fitness functions inspired by visual perception [5]. The current paper is motivated both by these earlier findings and by the gap in search-based PCG literature regarding graphical asset generation and visual quality evaluation.

2.2 Constrained Optimization and Constrained Novelty Search

Artificial evolution has traditionally been challenged in constrained spaces, as the division of evolving phenotypes between *feasible* (i.e. satisfying all constraints) and *infeasible* (i.e. failing one or more constraints) does not easily provide a gradient for search. A naive approach is to assign infeasible individuals the lowest fitness (*death penalty*), which has been argued against as infeasible individuals' valuable genetic information is lost [20]. Instead, a popular solution is to apply a penalty to the fitness of infeasible individuals [21]. Designing penalty functions can be challenging as a high penalty can become a death penalty while a low penalty can lead to extraneous exploration of infeasible space. Instead, [6] suggest that evolving infeasible individuals in a separate population to feasible ones circumvents the need to compare feasible with infeasible individuals. The suggested feasible-infeasible two-population genetic algorithm (FI-2pop GA) evolves an infeasible population towards minimizing the distance to feasibility while the feasible population evolves towards the problem-specific objective. Bringing infeasible individuals closer to the feasibility border increases the chances that they will produce feasible offspring; feasible offspring of infeasible individuals migrate to the feasible population and vice versa. The FI-2pop GA has been applied to search-based PCG for games, generating spaceships [22] and game levels [12, 14].

Novelty search [23] has recently been proposed as an alternative to objective-driven search. Novelty search prioritizes diversifying the population rather than gradually improving an objective score, and can outperform objective-driven search when the fitness function is deceptive, subjective, or unknown. Novelty search selects individuals based on their average distance to their nearest neighbors; these neighbors can be in the current population or in an archive of novel individuals. In each generation, novelty search may store the population's most novel individuals in this archive, which acts as a form of memory and promotes exploration of yet unvisited areas of the search space.

As with objective-driven search, novelty search can suffer from a search space divided into feasible and infeasible areas. Minimal-criteria novelty search applies the death penalty (a fitness of 0) to infeasible individuals [8], but can suffer from lost genetic information. Using a two-population approach, feasible-infeasible novelty search (FINS) evolves the feasible population according to novelty search, diversifying feasible individuals from the current population and from an archive of feasible individuals [7]. Since the two-population approach can apply different selection criteria on each

(a) (b) (c) (d) (e) (f)

Fig. 1: An example spaceship and its mutated offspring. The spaceship in Fig. 1a is constructed from the schema `UT15[RN01]UB12UB11[RN01RB08[UW02]DT02]UB05UC12[RW00]` which forms its genotype. The sprite itself is constructed via a turtle which places human-authored component sprites, a sample of which is shown in Fig. 1f (from top to bottom: `T15`, `W00`, `N01`, `B11` and `C12`). The remaining spaceships are produced from a single mutation of the spaceship in Fig. 1a. Fig. 1b applies the remove mutation; Fig. 1c applies the change sprite mutation. Fig. 1d applies the expand vertically mutation under the weapon attachments of the cockpit; Fig. 1e applies the expand horizontally mutation, as side sponsons left and right of the tail fins.

population, the feasible population diversifies its members via novelty search, while the infeasible population minimizes its members' distance from feasibility. Moreover, the feasible population creates an archive of novel individuals (containing only feasible results) to better explore the feasible search space. Feasible-infeasible dual novelty search (FI2NS) evolves both the feasible and the infeasible population towards novelty [7], and uses a different novel archive (with feasible and infeasible individuals respectively) for each population. Experiments with FINS and FI2NS in [24] have shown that FINS can quickly and reliably discover feasible individuals in highly constrained spaces, while FI2NS can create more diverse results in less constrained spaces.

3 Methodology

This paper uses constrained evolutionary approaches to generate spaceships constructed from multiple human-authored sprite components: the genetic encoding of these spaceships is detailed in Section 3.1. Section 3.2 describes the evolutionary algorithms, genetic operators and selection strategies used. Finally, Section 3.3 provides an overview of the constraints for plausible spaceships and the metrics for assessing visual style.

3.1 Spaceship Representation

The spaceships generated by the evolutionary algorithms are produced via turtle graphics, a drawing method where a cursor (turtle) moves upon a Cartesian plane. In this case, every step of the turtle equates to the placement of an image (i.e. a *sprite component*) taken from a library of human-authored images. The type of image and its filename is specified in the turtle command: the library contains multiple weapon sprites (`W`),

thruster sprites (`T`), body sprites (`B`), cockpit sprites (`C`), and connector sprites (`N`); see Fig. 1f for examples of each type. Each turtle command also specifies whether the sprite will be placed to the right (`R`), left (`L`), upwards (`U`) or downwards (`D`) of the current sprite. Each spaceship is thus encoded in multiple commands such as `RT01`, i.e. attach the thruster sprite (`T`) with ID `01` to the right of the current sprite. In order to allow for more complex, branching shapes, the `[` command pushes the current position and sprite of the cursor to a stack, while the `]` command restores the cursor to the previous position and sprite on the stack (removing it from the stack in the process).

The sequence of turtle commands which encode a spaceship is called a *schema*, and acts as the genotype for the evolutionary process. This schema is translated into a fully colored spaceship (see Fig. 1a); however, several constraints and fitness evaluations operate on the spaceship's *footprint* which treats all pixels of the spaceship as one color. The first command of a schema is placed at the center of a canvas: in this paper, the canvas has a preset size of 200 by 360 pixels. The schema describes only the right half of the spaceship: after running all the turtle's commands, spaceships are rendered symmetrical by reflecting the sprites horizontally, along the canvas center.

3.2 Spaceship Evolution

The genotype of every spaceship is an array of characters which act as commands (or parts of commands) for the turtle graphics. These genotypes are evolved via mutation alone: the mutation operators are chosen carefully in order to minimize the likelihood of infeasible or undesirable spaceships being created. An offspring is generated by iteratively applying mutation operators 1 to 4 times (chosen randomly) on the selected parent. There are four types of mutation possible in this evolutionary algorithm, as shown in Fig. 1b–1e: a mutation can remove one or more turtle commands, change the sprite ID of a turtle command, and expand a schema vertically or horizontally.

This paper tests spaceship evolution using constrained optimization (FI-2pop GA) and constrained novelty search (FINS) methods. Both methods use a two-population approach [6], where individuals which satisfy all constraints evolve in a different population than individuals which fail one or more constraints. In both FINS and FI-2pop GA, individuals in the infeasible population use objective-driven evolution, attempting to minimize their distance from feasibility which is aggregated from several constraints described in Section 3.3. The feasible population in FI-2pop GA evolves towards maximizing an objective pertaining to a specific visual style. For FINS, the feasible population evolves towards maximizing the novelty score ρ of eq. (1), which evaluates the average distance of an individual to its closest neighbors in the current population and in an archive of past novel discoveries. In all experiments in this paper, the 15 closest neighbors are considered when calculating novelty ($k = 15$), while in every generation the most novel feasible individual is added to the novel archive, which is initially empty.

$$\rho(i) = \frac{1}{k} \sum_{j=1}^{k} d(i, \mu_j) \tag{1}$$

where μ_j is the j-th-nearest neighbor of i (within the population and in the archive of novel individuals); distance $d(i, j)$ is a domain-dependent metric.

In all algorithms in this paper, parents are chosen via fitness-proportionate roulette wheel selection. The same parent can be chosen more than once, thus producing multiple mutated offspring. The fittest individual is carried over to the next generation (elitism of 1). Finally, the two evolving populations are artificially balanced via the *feasible offspring boost*, which is applied when the feasible population is smaller than the infeasible one: this mechanism assigns an equal number of offspring to both feasible and infeasible population (i.e. half of the total population) regardless of the number of parents in each. Although it depends on the chance of infeasible offspring from feasible parents, the feasible offspring boost generally increases the feasible population's size.

3.3 Spaceship Evaluation

Generated spaceships must fulfill certain criteria in order to be plausible. Spaceships which satisfy such constraints evolve to exhibit certain visual properties, while spaceships which do not satisfy one or more constraints are evaluated based on their distance to feasibility. Spaceships must satisfy the following constraints in order to be feasible:

- c_b Spaceships can not extend past the borders of the canvas (200 by 360 pixels).
- c_s Spaceships must be a single segment (measured via a 4-directional flood fill).
- c_o Spaceships can not have sprite components overlapping with other components
- c_w No sprite components can be in the weapons' line of fire.
- c_t No sprite components except connectors can be in the thrusters' path.
- c_m A spaceship's bounding box must not cover a surface smaller than 1024 pixels, as it will be too small for in-game use. Moreover, this constraint ensures that a feasible spaceship contains at least two sprite components.

Each of these constraints returns a score indicative of its distance from feasibility; only spaceships with a total distance of 0 for all constraints are feasible. Spaceships which satisfy all constraints are evaluated on their visual properties. Inspired by [5] and largely based on fitnesses implemented in [19], this paper evaluates primarily a spaceship's *balance*, operating either on its footprint or its bounding box. Thus, the spaceship is considered only in terms of its shape, ignoring its color; color may be considered in future work, however. This paper uses the following metrics:

- f_t The ratio between top half and bottom half of the spaceship's footprint (see Fig. 2a).
- f_{m_x} The ratio between the middle half of the spaceship's blueprint on the x axis (see Fig. 2b) and the remaining areas.
- f_{m_y} The ratio between the middle half of the spaceship on the y axis (see Fig. 2c) and the remaining areas.
- f_w The ratio between width and height of the spaceship's bounding box.
- f_{bb} The ratio between the surface of the spaceship's footprint and the surface of its bounding box.
- f_o The ratio between the spaceship's outline (see Fig. 2d) and the perimeter of its bounding box.
- f_c The total number of pixels between adjacent components which do not match (see Fig. 2e), normalized to the total pixels on the borders of adjacent components.

(a) f_t (b) f_{m_x} (c) f_{m_y} (d) f_o (e) f_c

Fig. 2: Evaluations of visual properties for the spaceship in Fig. 1a: f_t evaluates the surface ratio between the top half (red) and the bottom half (black); f_{m_x} evaluates the surface between the middle half of the spaceship, split horizontally (red), with the remaining areas (black); f_{m_y} evaluates the surface between the middle half of the spaceship, split vertically (red), with the remaining areas (black); f_o evaluates the ratio between the length of the spaceship's outline (red) and the perimeter of the bounding box (black frame); f_o evaluates whether connected sprite components (in different shades of gray) have imperfect connections (in red).

Except for f_c, all other metrics are not bound and have a value range of (0,∞), with a score of 1 indicating a balanced ratio (e.g. a square bounding box for f_w). Therefore all scores except f_c are normalized via the sigmoid of eq. (2) centered at $x = 1$. Due to the emphasis that $n(x)$ places on x values around 1 (e.g. slight imbalances between top and bottom halves in f_t), compared to small changes in $n(x)$ for large values of x, the normalized metrics are ideal for aggregating into a weighted sum for evolving spaceships towards multiple objectives of visual quality.

$$n(x) = \frac{1}{1 + e^{-5(x-1)}} \tag{2}$$

4 Experiments

In order to assess how evolution can achieve different visual styles in generated spaceships, the following indicative *style objective functions* are considered:

$$F_B = f_t + f_w + (1 - f_c) \tag{3}$$
$$F_O = f_o + f_{bb} + (1 - f_c) \tag{4}$$
$$F_H = f_{m_y} + f_{m_x} + (1 - f_c) \tag{5}$$
$$F_S = (1 - f_{m_y}) + (1 - f_t) + (1 - f_c) \tag{6}$$

Each of these objective functions describes a distinct visual style: F_B favors wide, front-loaded spaceships; F_O favors bulky, boxy spaceships (due to f_{bb}) which however have an interesting, complex outline (due to f_o); F_H favors spaceships which are aligned vertically and horizontally along the spaceship's centerpoint; finally, F_S favors spaceships with large side sponsons, while most of the main hull is located in the back. Note that all style objective functions attempt to minimize f_c; this fitness component is not a visual objective per se, but rather a soft constraint in order to deter tangential connections between sprite components which could lessen the plausibility of the spaceship.

While the functions of eq. (3)–(6) can be used as explicit objectives for a FI-2pop GA, novelty search requires a distance function to calculate its novelty score in eq. (1). Distance functions can be derived directly from the objectives of eq. (3)–(6) and target diversity along those specific visual properties. These *style distance functions* are shown in eq. (7)–(10), where Δf is the difference in scores of two spaceships for metric f.

$$D_B = \sqrt{(\Delta f_t)^2 + (\Delta f_w)^2 + (\Delta f_c)^2} \tag{7}$$

$$D_O = \sqrt{(\Delta f_o)^2 + (\Delta f_{bb})^2 + (\Delta f_c)^2} \tag{8}$$

$$D_H = \sqrt{(\Delta f_{m_y})^2 + (\Delta f_{m_x})^2 + (\Delta f_c)^2} \tag{9}$$

$$D_S = \sqrt{(\Delta f_{m_y})^2 + (\Delta f_t)^2 + (\Delta f_c)^2} \tag{10}$$

Novelty search using the distance functions of eq. (7)–(10) will likely result in different visual styles along those visual dimensions. However, evaluating difference in all visual metrics of Section 3.3 can prompt novelty search to explore a broader spectrum of possible spaceship designs. Using the distance function of eq. (11), novelty search can distinguish in a more granular fashion the visual differences between two spaceships, allowing it to simultaneously explore all combinations of visual metrics.

$$D_{all} = \sqrt{(\Delta f_t)^2 + (\Delta f_{m_x})^2 + (\Delta f_{m_y})^2 + (\Delta f_w)^2 + (\Delta f_{bb})^2 + (\Delta f_o)^2 + (\Delta f_c)^2} \tag{11}$$

In the experiments documented in the next section, four different FI-2pop GAs use the style objective functions of eq. (3)–(6) as their feasible fitness. Moreover, four different FINS algorithms use the distance functions of eq. (7)–(10) for novelty search on the feasible population; finally, a FINS algorithm identified as FINS$_{all}$ uses D_{all} of eq. (11) to diversify its feasible individuals. For all algorithms, the infeasible population minimizes the distance from feasibility for all constraints outlined in Section 3.3. Results are collected from 50 evolutionary runs, after 100 generations with a total population of 100 individuals per run (including feasible and infeasible ones). Significance reported in this paper uses a two-tailed t-test assuming unequal variance, with a significance level $\alpha = 0.05$; when performing multiple comparisons, the Bonferroni correction [25] is applied. In reported t-tests, normality is established via the Shapiro-Wilk test [26].

4.1 Quality of generated results

In order to evaluate the quality of generated spaceships, the fittest individual at the end of each evolutionary run is found using the style objective functions of eq. (3)–(6). The score of these individuals in the style objective and its visual metrics are shown in Table 1. While for FINS$_{all}$ results in the table are collected from the same set of 50 evolutionary runs, all other algorithms explicitly use the same objective and distance scores as the style objective which determines the best individuals (i.e. the best spaceships for F_O in Table 1 were collected from runs targeting F_O in FI-2pop GA and D_O in FINS).

Observing the results of Table 1, it is clear that for all visual styles explored in eq. (3)–(6) the objective-driven FI-2pop GA attains higher fitness scores than the constrained novelty search variants. In all cases the FI-2pop GA attains a significantly

F_B					
Method	Feasible	F_B	f_w	f_t	$1 - f_c$
FINS$_{all}$	41.3 (3.8)	0.89 (0.06)	0.94 (0.10)	0.97 (0.07)	0.77 (0.10)
FINS	44.3 (4.3)	0.87 (0.06)	0.89 (0.15)	0.94 (0.07)	0.76 (0.13)
FI-2pop GA	44.4 (4.3)	0.99 (0.01)	1.00 (0.01)	1.00 (0.00)	0.98 (0.03)
F_O					
Method	Feasible	F_O	f_o	f_{bb}	$1 - f_c$
FINS$_{all}$	41.3 (3.8)	0.80 (0.04)	0.92 (0.08)	0.66 (0.11)	0.82 (0.07)
FINS	42.9 (4.4)	0.79 (0.04)	0.85 (0.16)	0.68 (0.15)	0.84 (0.08)
FI-2pop GA	42.9 (5.2)	0.93 (0.01)	0.97 (0.02)	0.88 (0.04)	0.95 (0.04)
F_H					
Method	Feasible	F_H	f_{m_y}	f_{m_x}	$1 - f_c$
FINS$_{all}$	41.3 (3.8)	0.94 (0.02)	0.99 (0.01)	0.99 (0.01)	0.83 (0.06)
FINS	41.4 (4.0)	0.92 (0.03)	0.96 (0.06)	0.98 (0.04)	0.82 (0.08)
FI-2pop GA	46.8 (2.8)	0.99 (0.01)	1.00 (0.01)	1.00 (0.00)	0.98 (0.02)
F_S					
Method	Feasible	F_S	$1 - f_{m_y}$	$1 - f_t$	$1 - f_c$
FINS$_{all}$	41.3 (3.8)	0.80 (0.07)	0.76 (0.19)	0.88 (0.16)	0.77 (0.11)
FINS	43.0 (4.2)	0.81 (0.05)	0.82 (0.11)	0.85 (0.09)	0.77 (0.11)
FI-2pop GA	43.9 (3.9)	0.94 (0.02)	0.93 (0.04)	0.95 (0.02)	0.93 (0.05)

Table 1: Feasible individuals and best individuals' fitness scores at the end of each evolutionary run. Results are averaged from 50 independent runs with standard deviation in parentheses.

higher style objective score in its best individuals than both FINS and FINS$_{all}$. This is not surprising, since FI-2pop GA explicitly rewards a specific visual style, and the same heuristic is then used to find the best individuals. While the FI-2pop GA could potentially under-perform if the fitness landscape was deceptive, this apparently is not the case for the representation and mutation operators used and the fitnesses tested.

A more surprising finding of Table 1 is that the general novelty search of FINS$_{all}$ finds individuals of comparable (and in some cases higher) fitness as the more targeted FINS. Due to high deviation in both algorithms, only in experiments with F_B and F_H does FINS$_{all}$ show a statistically significant increase from FINS. This behavior is likely due to the fact that FINS$_{all}$ attempts to explore broadly, driven by a more granular distance function which differentiates spaceships based on many more visual properties than the style distance functions of eq. (7)–(10); through an indirect, serendipitous traversal of the search space it reaches areas of high scores in many visual metrics. How this exploration affects the diversity of results will be elaborated in Section 4.2.

Figure 3 shows the fittest individuals among the 50 independent runs for the different style objectives and algorithms. Different sprite colors are used to highlight the different style objectives. All evolutionary methods create spaceships which exhibit the intended visual properties of the aggregated fitness functions. For F_O, the best spaceships are boxy; for F_B they are wide and very elongated, and the bottom of the spaceship contains only thrusters; for F_H they are more "centrally" located around the image's center, with most spaceship components extending vertically and horizontally from that

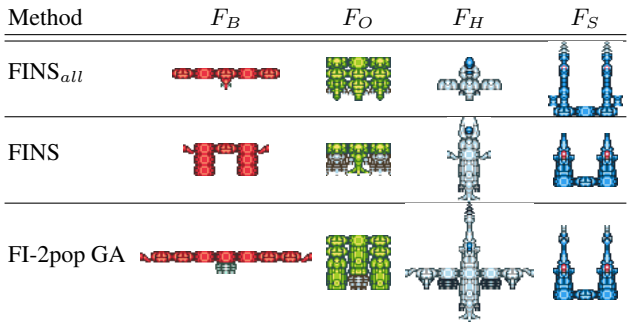

Fig. 3: Overall fittest spaceships in 50 independent runs per approach and fitness function.

point; for F_S they have sizable side-sponsons and a smaller main hull at the bottom of the spaceship. While all style objectives reward a low f_c score, this is difficult to pinpoint in the best spaceships: as noted when defining the style objectives, minimizing f_c acts as a control mechanism rather than an explicit visual goal; only cases with high f_c would make its presence obvious as spaceships would appear almost disjointed.

4.2 Diversity of generated results

The diversity of generated results is evaluated in this paper from the perspective of visual properties identified in Section 3.3 and particularly the visual styles included in eq. (3)–(6), rather than a general visual difference. Ignoring the f_c dimension as it is a control mechanism with little visual impact, all the spaceships collected by the end of each evolutionary approach are evaluated on the remaining two dimensions of eq. (3)–(6) and plotted on Fig. 4. As with experiments in Section 4.1, spaceships from the same 50 evolutionary runs for FINS$_{all}$ are used for all plots, while spaceships for FINS and FI-2pop GAs explicitly use the same style distance and style objective as the metrics plotted (i.e. for the F_O plot, spaceships are collected from 50 independent runs of FINS using D_O as its distance function and FI-2pop GA using F_O as its objective).

Observing Fig. 4, most spaceships evolved via the FI-2pop GA unsurprisingly have high scores in both visual dimensions of the style objective function being optimized (with most points concentrated at the top right corner of each plot). Evolution towards each style objective creates a different spread, however, as results for F_H are much more concentrated near (1,1) than for F_S, while for F_O most spaceships have high scores in f_o but not particularly high scores in f_{bb}. To a degree, this is due to the fact that no sprite component fully covers its own bounding box, but also due to the fact that "boxy" spaceships of high f_{bb} tend to have simple outlines (f_o); the conflicting nature of these metrics allows f_o to dominate evolution as it is easier to optimize.

For FINS and FINS$_{all}$, the space of the two visual dimensions is more broadly explored, with interesting differences in the spread of results depending on both the approach and the visual dimensions. For instance, while for F_B a very broad range of f_t and f_w values are discovered, for F_O only average scores in both f_o and f_{bb} are discovered. Both novelty search approaches tend to concentrate on specific areas of the

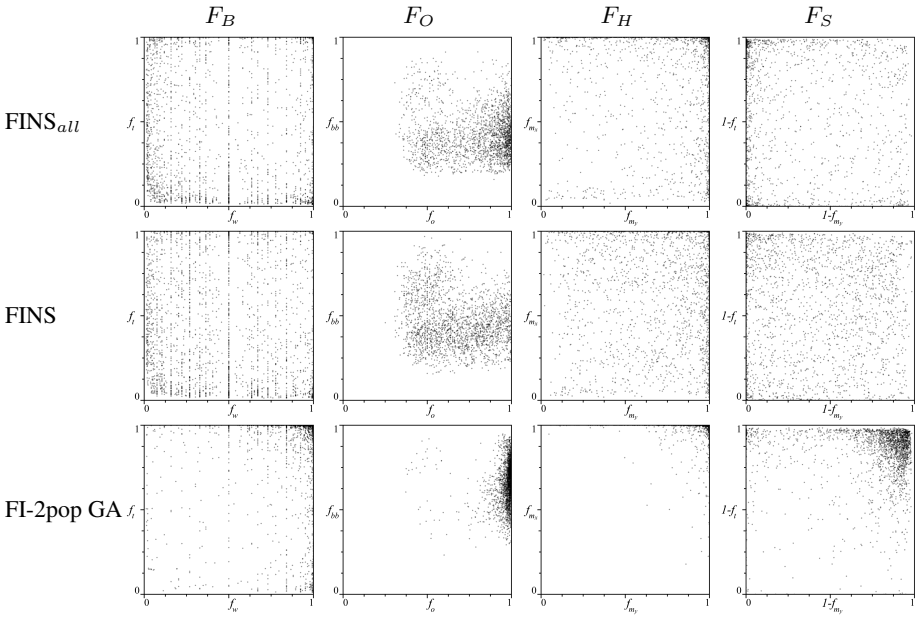

Fig. 4: Scatter plots of the visual metrics of all final spaceships in 50 independent runs, per evolutionary process and fitness function.

search space while exploring others to a smaller extent: for F_B more spaceships have low scores in f_t and f_w than high scores (while many spaceships also have $f_w = 0.5$, i.e. spaceships with square bounding boxes), for F_S more spaceships have low scores for $1 - f_{m_y}$, and (unsurprisingly) high scores in f_{m_y} and/or f_{m_x} for F_H. The bias towards certain areas of the search space is largely due to the representation and mutation operators used: due to the initial centrally placed sprite component and mutation operators which expand it vertically and horizontally, spaceships often even serendipitously receive high f_{m_y} and f_{m_x} scores, which is the case for both F_S and F_H. A visual inspection of Fig. 4 suggests that the targeted FINS explores more of the space of those dimensions included in its visual distance function than FINS$_{all}$, which explores more visual properties not included in the plot; this observation is tested numerically next.

In order to evaluate the diversity of results more methodically, each of the 50 independent runs is analyzed individually: the nearest neighbor distance of all spaceships of the same run is averaged and included in Table 2. Moreover, the final spaceships of the same run are clustered via k-medoids to produce six clusters[1]. The average inter-cluster distance (i.e. the mean distance between all clusters' medoids), the average intra-cluster distance (i.e. the mean distance between all members of the same cluster) and the average size of the cluster are used to evaluate the clusters' quality. Finally, the clusters are evaluated via the Dunn index [27], which identifies how compact each cluster is and how separated different clusters are (by dividing the shortest inter-cluster distance by

[1] Due to stochasticity, the chosen clusters are the best of 100 independent clustering attempts.

	nearest neighbor distance	inter-cluster distance	intra-cluster distance	cluster size	Dunn index
F_B					
FINS$_{all}$	0.065 (0.009)	0.700 (0.078)	0.124 (0.018)	6.9 (0.6)	2.26 (0.65)
FINS	0.068 (0.011)	0.639 (0.084)	0.135 (0.013)	7.4 (0.7)	2.01 (0.57)
FI-2pop GA	0.041 (0.013)	0.626 (0.108)	0.07 (0.025)	7.4 (0.7)	3.24 (1.09)
F_O					
FINS$_{all}$	0.037 (0.007)	0.329 (0.062)	0.059 (0.012)	6.9 (0.6)	2.13 (0.74)
FINS	0.043 (0.006)	0.331 (0.050)	0.071 (0.009)	7.2 (0.7)	1.88 (0.48)
FI-2pop GA	0.020 (0.007)	0.224 (0.080)	0.028 (0.009)	7.1 (0.9)	2.96 (1.47)
F_H					
FINS$_{all}$	0.049 (0.011)	0.655 (0.092)	0.088 (0.015)	6.9 (0.6)	2.78 (0.88)
FINS	0.063 (0.011)	0.594 (0.091)	0.117 (0.015)	6.9 (0.7)	2.11 (0.71)
FI-2pop GA	0.017 (0.007)	0.299 (0.115)	0.022 (0.013)	7.8 (0.5)	4.3 (4.55)
F_S					
FINS$_{all}$	0.056 (0.013)	0.681 (0.078)	0.104 (0.02)	6.9 (0.6)	2.44 (0.88)
FINS	0.069 (0.010)	0.606 (0.079)	0.127 (0.014)	7.2 (0.7)	1.87 (0.48)
FI-2pop GA	0.044 (0.015)	0.548 (0.129)	0.064 (0.024)	7.3 (0.7)	3.39 (1.27)

Table 2: Diversity and cluster quality of the final spaceships per evolutionary approach and fitness function. Results are averaged from 50 independent runs with standard deviation in parentheses.

the longest intra-cluster distance). All of these metrics are included in Table 2. When calculating distances and for clustering, only the first two visual dimensions of the style objective functions of eq. (3)–(6) are considered; f_c is omitted as it does not have much visual impact and as its value range is similar in all experiments.

Unsurprisingly, the FI-2pop GA has a significantly lower nearest neighbor distance than both FINS and FINS$_{all}$, which is obvious from Fig. 4. For similar reasons, the cluster medoids of spaceships evolved via FI-2pop GA are significantly closer together than those in FINS and FINS$_{all}$ (except in F_B). Surprisingly, the cluster quality based on the Dunn index is significantly higher for FI-2pop GA than for either novelty search approach; while the inter-cluster distance is relatively low, the intra-cluster distance is even lower (as all individuals are found in the same area of the search space, especially for F_H). Despite the fact that the clusters in the FI-2pop GA are not very separated, they are compact. Finally, regarding the clusters' size, all three evolutionary approaches had similarly sized clusters: many runs resulted in single-member clusters, as well as large ones. Clusters for the FI-2pop GA had more extreme size deviations, likely due to lower-quality outliers (caused by mutation) creating single-member clusters.

Comparisons between FINS and FINS$_{all}$ yield more interesting results. The dispersed appearance of FINS plots in Fig. 4 is validated via the nearest neighbor distance metric, which is significantly higher than FINS$_{all}$ in all cases except F_B. However, the k-medoid clusters for FINS$_{all}$ have a significantly higher inter-cluster distance than FINS in all cases except F_O. Individual spaceships are further dispersed with FINS (which rewards diversity along the same axes being evaluated), as evidenced by a high nearest neighbor distance and high intra-cluster distance, but the broader exploration

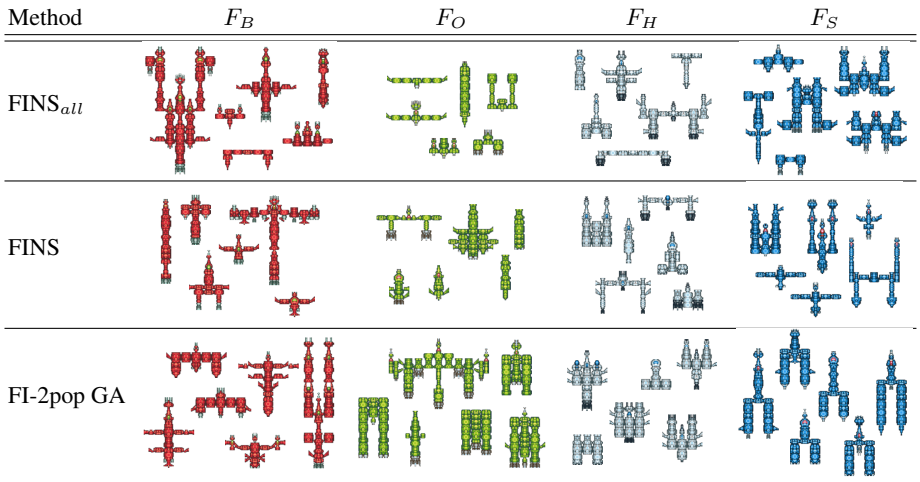

Fig. 5: Cluster medoids of the independent runs with the highest inter-cluster distance (out of 50).

of FINS$_{all}$ via a more granular distance function results in more easily separable and concise clusters, as evidenced by a high inter-cluster distance and a high Dunn index.

Figure 5 shows the cluster medoids of the run with the highest inter-cluster distance. The cluster medoids for the FI-2pop GA appear more similar; most medoids exhibit the intended visual style specified in the style objective, although major exceptions are the F_B objective which includes spaceships with wide, narrow and square bounding boxes as well as F_O which includes spaceships which do not cover much of the bounding box but exhibit complex outlines. Even so, cluster medoids of the FI-2pop GA exhibit at least one (and often both) visual properties in the objective, pointing to an overall consistent visual style in their results. On the other hand, medoids of either FINS or FINS$_{all}$ are visually very distinct (with few exceptions); it is important to note that even so, the medoids are different from one style objective to the next as well, confirming the impact that the visual metrics have on clustering. Taking F_S as an example, some medoids of FINS and FINS$_{all}$ have large side-sponsons while others are triangular, centrally aligned (horizontally and vertically), or front-loaded. Some medoids are more similar to each other, while others are unique. Overall, the cluster medoids for the different approaches are an indication of the range of visual styles which can be achieved via constrained novelty search, as well as the visual consistency (in most cases) which can be accomplished with the FI-2pop GA.

5 Discussion

Experiments in this paper evaluated the performance of two-population constrained evolutionary methods at generating game content of a designer-specified visual style as well as content of different, distinct visual styles. The objective-driven FI-2pop GA is ideal for creating spaceships of a visual style specified as an objective function, and the vast majority of its generated results possess the desired visual properties. However,

in many cases the single-minded search of the FI-2pop GA can create "bizarre" spaceships which over-optimize the objectives (leading e.g. to the extremely wide spaceships for F_B in Fig. 3). Novelty search applied to feasible individuals creates visually different spaceships, yet there is no guarantee that most of those will score highly in a style objective. Surprisingly, using a more inclusive measure of visual diversity for novelty search can lead to higher scores in certain visual metrics than when novelty search explicitly diversifies those dimensions. Finally, the quality and diversity of results depends on the visual metrics: for certain combinations (such as F_O) optimization is dominated by one dimension while novelty search can not explore all of the search space.

This initial attempt at game content generation focused on the spaceships' shape, disregarding other characteristics. Color was disregarded since spaceships are built from sprite components of a single hue (depending on the style objective); evaluating color would not have much impact. Color evaluations (via e.g. histogram analysis) would be an important addition if sprites of multiple colors could be added to the same spaceship. Among the current metrics, the ratio of the spaceship's surface could be evaluated in a more refined way via a center of mass rather than by dividing the spaceship in half. Finally, more data-driven approaches could be used to derive visual metrics and distance functions (replacing or complementing the current ones which are based on cognitive psychology theory), such as object detection libraries for computer vision, e.g. in [28], or deep learning methods trained on previously generated content, e.g. in [29].

Results in this paper show that the generator's expressivity and the visual range of resulting content is sensitive to the design of both the visual metrics and the mutation operators. As noted in Section 3.2, the mutation operators were carefully crafted to avert infeasible spaceships, but also bias generation towards specific styles such as the centrally aligned spaceships of F_H. This affects novelty search which does not explore the search space as broadly as intended. Additionally, most visual metrics needed to be normalized via eq. (2), as initial experiments with aggregated metrics in a style objective function indicated that some metrics tended to dominate others. These design choices, coupled with certain ad-hoc decisions (e.g. splitting the spaceship in halves) could affect the conclusions drawn from this study. Future work should improve and expand the mutation operators and methodically test the visual metrics.

While this paper focused almost exclusively on the visual style of the generated content, driven in part by a lack in the PCG literature in visual quality assessment, the functional properties of these spaceships should also be considered in future work. Since the generated spaceships could be used as enemies for an arcade-style game such as *Raptor: Call of the Shadows* (Apogee 1994), functional concerns could include how well the spaceships can fly or how dangerous they are. Currently, functional properties are considered only in constraints c_t and c_w, which ensure that the spaceship can fly and shoot; however, additional constraints on a minimum number of weapon components or a minimum and maximum speed for the spaceship would ensure that spaceships without thrusters (as in instances of Fig. 3) do not find their way into a game. For feasible spaceships, additional evaluations of their in-game performance (and game balance) would require simulations in a game engine, and could maximize their performance in pre-determined tasks (e.g. flying, avoiding obstacles, chasing enemies), similar to [22].

6 Conclusion

This paper described a method for evolving game content via turtle graphics where human-authored sprites are combined to create a complete, computer-generated spaceship. Generated results are evaluated on plausibility constraints and on visual metrics of the spaceship shape's balance and complexity. Two constrained evolutionary approaches, a feasible-infeasible two population genetic algorithm and feasible-infeasible novelty search, were tested on whether they generate game content of a specified visual style as well as content of different, distinct visual styles. Results indicate that while objective-driven search can create spaceships with a consistent visual style, the enhanced exploration of visual metrics afforded by novelty search can lead to interesting content which do not suffer from over-optimizing the engineered metrics of visual quality. Future work could improve the visual metrics, potentially replacing them with data-driven models, and include evaluations of the playability of generated content.

Acknowledgements

The sprite components used to construct the spaceships are freely licensed art assets found in OpenGameArt (`http://opengameart.org/content/modular-ships`) and are not the intellectual property of the author. The research was supported, in part, by the FP7 Marie Curie CIG project AutoGameDesign (project no: 630665).

References

1. Liapis, A., Yannakakis, G.N., Togelius, J.: Computational game creativity. In: Proceedings of the Fifth International Conference on Computational Creativity. (2014)
2. Champandard, A.: EA investing in the next-generation with proceduralism post-SSX. `http://aigamedev.com/open/teaser/investing-in-procedural/` (May 2012) [Online; accessed 11-January-2016].
3. Blizzard Entertainment: What is Diablo III? `http://us.battle.net/d3/en/game/what-is` [Online; accessed 11-January-2016].
4. Togelius, J., Nelson, M.J., Liapis, A.: Characteristics of generatable games. In: Proceedings of the FDG Workshop on Procedural Content Generation. (2014)
5. Arnheim, R.: Art and visual perception: a psychology of the creative eye. Revised and expanded edn. University of California Press (2004)
6. Kimbrough, S.O., Koehler, G.J., Lu, M., Wood, D.H.: On a feasible-infeasible two-population (FI-2Pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. European Journal of Operational Research **190**(2) (2008)
7. Liapis, A., Yannakakis, G.N., Togelius, J.: Enhancements to constrained novelty search: Two-population novelty search for generating game content. In: Proceedings of Genetic and Evolutionary Computation Conference. (2013)
8. Lehman, J., Stanley, K.O.: Revising the evolutionary computation abstraction: Minimal criteria novelty search. In: Proceedings of the Genetic and Evolutionary Computation Conference. (2010)
9. Preuss, M., Liapis, A., Togelius, J.: Searching for good and diverse game levels. In: Proceedings of the IEEE Conference on Computational Intelligence and Games (CIG). (2014)

10. Smith, G., Whitehead, J.: Analyzing the expressive range of a level generator. In: Proceedings of the FDG workshop on Procedural Content Generation. (2010)
11. Togelius, J., Yannakakis, G.N., Stanley, K.O., Browne, C.: Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games **3**(3) (2011)
12. Liapis, A., Yannakakis, G.N., Togelius, J.: Towards a generic method of evaluating game levels. In: Proceedings of the AAAI Artificial Intelligence for Interactive Digital Entertainment Conference. (2013)
13. Lara-Cabrera, R., Cotta, C., Leiva, A.J.F.: A procedural balanced map generator with self-adaptive complexity for the real-time strategy game planet wars. In: Proceedings of Applications of Evolutionary Computation. Volume 7835, LNCS. Springer (2013)
14. Sorenson, N., Pasquier, P., DiPaola, S.: A generic approach to challenge modeling for the procedural creation of video game levels. IEEE Transactions on Computational Intelligence and AI in Games **3**(3) (2011)
15. Liapis, A., Yannakakis, G.N., Togelius, J.: Sentient Sketchbook: Computer-aided game level authoring. In: Proceedings of the 8th Conference on the Foundations of Digital Games. (2013)
16. Howlett, A., Colton, S., Browne, C.: Evolving pixel shaders for the prototype video game subversion. In: Proceedings of the AI and Games Symposium (AISB'10). (2010)
17. Hastings, E.J., Guha, R.K., Stanley, K.O.: Automatic content generation in the galactic arms race video game. IEEE Transactions on Computational Intelligence and AI in Games **1**(4) (2009)
18. Risi, S., Lehman, J., D'Ambrosio, D., Hall, R., Stanley, K.O.: Combining search-based procedural content generation and social gaming in the petalz video game. In: Proceedings of Artificial Intelligence and Interactive Digital Entertainment Conference. (2012)
19. Liapis, A., Yannakakis, G.N., Togelius, J.: Adapting models of visual aesthetics for personalized content creation. IEEE Transactions on Computational Intelligence and AI in Games **4**(3) (2012)
20. Michalewicz, Z.: Do not kill unfeasible individuals. In: Proceedings of the Intelligent Information Systems Workshop. (1995)
21. Coello Coello, C.A.: Constraint-handling techniques used with evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, ACM (2010)
22. Liapis, A., Yannakakis, G.N., Togelius, J.: Neuroevolutionary constrained optimization for content creation. In: Proceedings of the IEEE Conference on Computational Intelligence and Games. (2011)
23. Lehman, J., Stanley, K.O.: Abandoning objectives: Evolution through the search for novelty alone. Evolutionary Computation **19**(2) (2011)
24. Liapis, A., Yannakakis, G.N., Togelius, J.: Constrained novelty search: A study on game content generation. Evolutionary Computation **23**(1) (2015)
25. Dunn, O.: Multiple comparisons among means. Journal of the American Statistical Association (2012)
26. Shapiro, S.S., Wilk, M.B.: Analysis of variance test for normality (complete samples). Biometrika (1965)
27. Dunn, J.C.: A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Journal of Cybernetics (3) (1973)
28. Correia, J., Machado, P., Romero, J., Carballal, A.: Evolving figurative images using expression-based evolutionary art. In: Proceedings of the International Conference on Computational Creativity. (2013)
29. Liapis, A., Martínez, H.P., Togelius, J., Yannakakis, G.N.: Transforming exploratory creativity with DeLeNoX. In: Proceedings of the International Conference on Computational Creativity. (2013)